Sweetening the sour taste of inhomogeneous signature-based Gröbner basis computations

Christian Eder
c/o Department of Mathematics
TU Kaiserslautern
67653 Kaiserslautern, Germany
ederc@mathematik.uni-kl.de

Abstract

In this paper we want to give an insight in the rather unknown behaviour of signature-based Gröbner basis algorithms, like F5, G2V, or GVW, for inhomogeneous input. On the one hand, it seems that the restriction to sig-safe reductions in those algorithms puts a huge penalty on their performance. The lost connection between polynomial degree and signature degree can disallow lots of reductions and lead to a huge overhead in the computations. On the other hand, the way critical pairs are sorted and the corresponding s-polynomials are handled is a quite good one. We show in detail the strong connection to the sorting of critical pairs w.r.t. well-known sugar degree of polynomials. Those properties hold for signature-based Gröbner basis algorithms in general, not depending on specific implementations of the underlying criteria to discard useless critical pairs.

1 Introduction

Computing Gröbner bases is a fundamental tool in commutative and computer algebra. Buchberger introduced the first algorithm to compute such bases in 1965, see [6]. In the meantime lots of additional and improved algorithms have been developed.

In the last couple of years, so-called *signature-based* algorithms like F5, see [11], and G2V, see [13], have become more popular. Lots of optimizations for these algorithms have been published, for example, see [9, 1, 14, 19, 21]. Whereas the above mentioned publications focus on the area of optimizing signature-based criteria for detecting useless critical pairs, a close look at the overall behaviour of signature-based computations in general is still missing. Here we want to fill this gap and discuss advantages and disadvantages of the signature-based attempt: Without going into detail about efficient implementations we analyze the underlying characteristics all of the corresponding algorithms share:

1. Sorting critical pairs by increasing signatures, and
2. processing only sig-safe reduction steps.

By doing this we clear up myths like signature-based algorithms are only applicable for homogeneous input data, but they are not useful (either in the sense of being incorrect or in the sense of being under-performing) in the inhomogeneous setting.

In Section 2 we introduce the basic setting for signature-based Gröbner basis algorithms. There we unify the fundamental framework for all such algorithms, explaining in detail the differences to a pure polynomial approach. Making some smaller changes to the initial presentation of F5 in Section 3, we enable it to compute Gröbner bases for inhomogeneous input, too. Even more, it turns out that with this new description understanding the algorithm's inner workings is much easier. Following this, we give for the first time a discussion about the strong connections between the sorting of critical pairs by increasing signatures and the corresponding sorting by the so-called sugar degree, which is introduced in [16] and further discussed in [3]. The sugar degree is known to be a powerful tool optimizing pure polynomial Gröbner basis computations in the inhomogeneous setting. Thus, explaining its relation to the signature-based world, we are able to allow a first estimate for the usefulness of those kind of algorithms beyond the homogeneous case. Even more, in Section 5 we test the differences in the behaviour of sig-safe reductions comparing 4 different implementations of signature-based algorithms for a wide range of examples side-by-side in the respective inhomogeneous and homogenized version. It turns out that there is, compared to pure polynomial Gröbner basis algorithms speaking in terms of inhomogeneous computations, no such built-in disadvantage observable signature-based algorithms rely on.

The main contribution of this paper is to give a deeper insight in the inner workings of signature-based Gröbner basis algorithms with a view towards optimizing the order in which critical pairs are handled. Besides this, we give first ideas for good heuristics to decide when to use which variant of the known algorithms in order to benefit from a better performance.

2 Basic setting

Let us start with some basic notations. Let $i \in \mathbb{N}$, $\mathcal{K}$ a field, and $\mathcal{R} = \mathcal{K}[x_1, \ldots, x_n]$. Furthermore, we denote the monoid of all monomials in $x_1, \ldots, x_n$ by $\mathcal{M} := \{\prod_{i=1}^n x_i^{\alpha_i} \mid (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n\}$. We mostly use the shorthand notation $x^\alpha := \prod_{i=1}^n x_i^{\alpha_i}$. A polynomial $p \in \mathcal{R}$ is a finite $\mathcal{K}$-linear combination of monomials in $\mathcal{R}$,

$$p = \sum_{\alpha \in \mathbb{N}^n}^{\text{finite}} c_\alpha x^\alpha, c_\alpha \in \mathcal{K}.$$

We define the **degree of a polynomial** $p \neq 0$[1] by

$$\deg(p) = \max \left\{ \sum_{i=1}^n \alpha_i \mid c_\alpha \neq 0 \right\}.$$

We say that a polynomial p is **homogeneous**, if all its monomials have the same degree; otherwise we call p **inhomogeneous**.

[1] For the zero polynomial we set $\deg(0) = -1$.

Let $F = (f_1, \ldots, f_m)$, where each $f_i \in \mathcal{R}$, and $I = \langle F \rangle \subset \mathcal{R}$ is the ideal generated by the elements of F.

Moreover, if we fix a well-ordering $<$ on the monoid $\mathcal{M}$ of monomials of $x_1, \ldots, x_n$ we get a unique representation of the elements in $\mathcal{R}$: For a polynomial $p \in \mathcal{R}$, we denote p's **leading monomial** by $\operatorname{lm}(p)$, its **leading coefficient** by $\operatorname{lc}(p)$, and write $\operatorname{lt}(p) = \operatorname{lc}(p) \operatorname{lm}(p)$ for its **leading term**. In particular, a well-ordering $<$ prefering the degree over any other criterion to sort elements is denoted **degree-prefering ordering**. For any two polynomials $p, q \in \mathcal{R}$ we use the shorthand notation $\tau(p, q) := \operatorname{lcm}(\operatorname{lm}(p), \operatorname{lm}(q))$ for the **least common multiple** of their leading monomials.

Let $e_1, \ldots, e_m$ be the canonical generators of the free $\mathcal{R}$-module $\mathcal{R}^m$.We define a map

$$\begin{array}{rccc} \pi: & \mathcal{R}^m & \to & I \\ & \sum_{i=1}^m p_i e_i & \mapsto & \sum_{i=1}^m p_i f_i, \end{array}$$

such that the p_i's are polynomials in $\mathcal{R}$. Thus we extend the ordering $<$ to an admissible ordering $\prec$ on the set $\mathcal{M}' := \{te_i \mid t \in \mathcal{M}, 1 \leq i \leq m\}$. Without any restriction, the reader can think of the following two choices for $\prec$ in the following:

1. prefering the module position over the term $\prec_{\text{pot}}$:
   $t_i e_i \prec t_j e_j$ iff $i < j$, or $i = j$ and $t_i < t_j$.
2. Being induced by $<$, the Schreyer ordering $\prec_{\text{s}}$:
   $t_i e_i \prec t_j e_j$ iff $t_i \operatorname{lm}(\pi(e_i)) < t_j \operatorname{lm}(\pi(e_j))$, or $t_i \operatorname{lm}(\pi(e_i)) = t_j \operatorname{lm}(\pi(e_j))$ and $i < j$.

In [**?**] it is shown that these two orderings are the most efficient ones for signature-based Gröbner basis computations nowadays. The author has made the very same experiences in different tests of his implementations.

Since most of our considerations in this paper are independent of the chosen extended ordering we mostly use the notation $\prec$ and specify to $\prec_{\text{pot}}$ respectively $\prec_{\text{s}}$ whenever differences appear. Clearly, the notions of leading monomial, leading term, and leading coefficient generalize naturally to $\mathcal{R}^m$ w.r.t. $\prec$ on $\mathcal{M}'$.[2]

An element $\omega \in \mathcal{R}^m$ with $\pi(\omega) = 0$ is called a **syzygy of** $f_1, \ldots, f_m$. The module of all such syzygies is denoted $\operatorname{Syz}(F)$. A syzygy of type $f_j e_i - f_i e_j$ is called a **principal syzygy**. The set of all principal syzygies of F is denoted $\operatorname{PSyz}(F)$; clearly, $\operatorname{PSyz}(F) \subseteq \operatorname{Syz}(F)$. Note that if a sequence F of polynomials is regular, then $\operatorname{PSyz}(F) = \operatorname{Syz}(F)$.

Definition 2.1. *Let $p \in I$ and let $h_1, \ldots, h_m \in \mathcal{R}, h \in \mathcal{R}^m$ such that $p = \pi(h)$ where $h := \sum_{i=1}^m h_i e_i$.*

We say that $\operatorname{lm}(h)$ is **a signature** *of p. Clearly, a signature of p is always an element of $\mathcal{M}'$. Moreover, considering the well-ordering $\prec$ on $\mathcal{M}'$ there exists for each $p \in \mathcal{R}$ a unique, minimal signature.*

From this point of view it makes sense to equip polynomials in I with a corresponding signature. For an easier description in the following let us agree on the notation $\mathcal{L} := \mathcal{M}' \times I$.

[2] In the following it is always clear from the context w.r.t. which ordering, $<$ respectively $\prec$, leading elements are considered. Thus we can abandon any distinction for the corresponding notations.

Definition 2.2. *Let $p \in I$.*

1. *An element $f = (te_i, p) \in \mathcal{L}$ is called a* **labeled polynomial**, *if te_i is a signature of p. For a labeled polynomial $f = (te_i, p)$ we define the shorthand notations $\mathrm{poly}(f) = p$, $\mathrm{sig}(f) = te_i$, and $\mathrm{index}(f) = i$. Talking about the leading monomial, leading term, leading coefficient, degree, and least common multiples of labeled polynomials f and g we always assume the corresponding value of $\mathrm{poly}(f)$ and $\mathrm{poly}(g)$. Furthermore, if $G = \{g_1, \ldots, g_\ell\} \subset \mathcal{L}$, then we define $\mathrm{poly}(G) := \{\mathrm{poly}(g_1), \ldots, \mathrm{poly}(g_\ell)\} \subset I$.*

2. *Let $f \in \mathcal{L}$, let $t \in \mathcal{M}$, and let $c \in \mathcal{K}$. We define a multiplication of f by ct:*[3]
$$ctf := (t\mathrm{sig}(f), ct\,\mathrm{poly}(f)) \in \mathcal{L}.$$

3. *A* **critical pair** *of two labeled polynomials f and g is a tuple $(f, g) \in \mathcal{L}^2$. The* **degree of a critical pair** *is defined by*
$$\deg(f, g) := \deg\left(\tau(f, g)\right).$$
Moreover, we define the **s-polynomial** *of two labeled polynomials f and g in $\mathcal{L}$ by*
$$\mathcal{S}(f, g) = \left(\omega, u_f \cdot \mathrm{poly}(f) - \frac{\mathrm{lc}(f)}{\mathrm{lc}(g)} u_g \cdot \mathrm{poly}(g)\right)$$
where $u_f = \frac{\tau(f,g)}{\mathrm{lm}(f)}$, $u_g = \frac{\tau(f,g)}{\mathrm{lm}(g)} \in \mathcal{M}$, and $\omega = \mathrm{lm}\left(u_f \mathrm{sig}(f) - u_g \mathrm{sig}(g)\right)$. Furthermore, an s-polynomial $\mathcal{S}(f, g)$ fulfilling that
$$\omega \prec \max\left\{u_f \mathrm{sig}(f), u_g \mathrm{sig}(g)\right\}$$
is called **non-minimal**.

4. *We define the* **signature degree** *of a labeled polynomial $f = (te_i, p) \in \mathcal{L}$ by*
$$\text{sig-deg}(f) := \deg(t) + \deg\left(\pi(e_i)\right).$$
Moreover, in the following it makes sense to speak about the signature degree of a critical pair: $\text{sig-deg}(f, g) := \text{sig-deg}\left(\mathcal{S}(f, g)\right)$.

On our way giving a basic frame for signature-based Gröbner basis algorithms it is left to extend the notions of reduction and standard representation from the pure polynomial setting to the labeled one:

Definition 2.3. *Let $f, g \in \mathcal{L}$ be two labeled polynomials. Moreover, let $G \subset \mathcal{L}$ with $\#G = k$.*

1. *We say that f* **reduces sig-safe to** *g* **modulo** *G if there exist $r_0 = f, \ldots, r_k = g \in \mathcal{L}$ such that for all $i \in \{1, \ldots, k\}$ there exist $g_{j_i} \in G$, $t_i \in \mathcal{M}$, and $c_i \in \mathcal{K}$ fulfilling*

   (a) $r_i = r_{i-1} - c_i t_i g_{j_i}$,

[3] In [10] it is shown that one does not need to consider signatures as leading terms of elements in $\mathcal{R}^m$, the monomial data is sufficient.

(b) $\operatorname{lm}(r_i) < \operatorname{lm}(r_{i-1})$, *and*

(c) $t_i \operatorname{sig}(g_{j_i}) \prec \operatorname{sig}(r_{i-1})$.

2. *f has a* **standard representation with respect to** *G if there exist* $h_1, \ldots, h_k \in \mathcal{R}$, $g_1, \ldots, g_k \in G$ *such that* $\operatorname{poly}(f) = \sum_{i=1}^{k} h_i \operatorname{poly}(g_i)$, *and for each* $i \in \{1, \ldots, k\}$ *either* $h_i = 0$, *or*

   (a) $\operatorname{lm}(h_i) \operatorname{lm}(g_i) \leq \operatorname{lm}(f)$, *and*

   (b) $\operatorname{lm}(h_i) \operatorname{sig}(g_i) \preceq (f)$.

3. *If there exists* $h \in G$ *such that* $\operatorname{sig}(h) \mid \operatorname{sig}(f)$ *and* $\operatorname{lm}(h) \mid \operatorname{lm}(f)$, *then we say that f* **is sig-redundant to** *G.*

Remark 2.4. *If f reduces sig-safe to g modulo G, then it has a standard representation modulo G. Moreover, note that the concept of sig-safeness, that means the restriction of the reducer g_{j_i} by $t_i \operatorname{sig}(g_{j_i}) \prec \operatorname{sig}(r_{i-1})$ in each step, is essential for the correctness (and the performance) of signature-based algorithms.*

Clearly, if a labeled polynomial f has a standard representation w.r.t. G, then $\operatorname{poly}(f)$ has a standard representation w.r.t. $\operatorname{poly}(G)$. Thus it is no wonder that we can give a statement similar to Buchberger's Criterion, see [6], for the signature-based setting.

Theorem 2.5. *Let* $G = \{g_1, \ldots, g_k\} \subset \mathcal{L}$ *such that* $\{f_1, \ldots, f_m\} \subset \operatorname{poly}(G)$. *If for each pair* (i, j) *with* $i > j$, $1 \leq i, j \leq k$, *either*

1. $\mathcal{S}(g_i, g_j)$ *is non-minimal, or*

2. $\mathcal{S}(g_i, g_j)$ *has a standard representation w.r.t. G,*

then $\operatorname{poly}(G)$ *is a Gröbner basis of I.*

Proof. See, for example, [9, 10]. □

Remark 2.6. *The first property in Theorem 2.5 seems quite strange from a purely polynomial point of view. In several other publications it is already shown that non-minimal elements are useless for the resulting Gröbner basis as well as for the algorithm. Thus, they need not be considered at all. We refer to [10] for more details on this peculiarity of signature-based algorithms.*

Let us have a look at a quite generic signature-based Gröbner basis algorithm. Note that our emphasis lies on the explanation of the general idea behind signature-based computations and not on the presentation of pseudo code of efficient implementations.

Clearly, as in the polynomial-only setting a Gröbner basis algorithm without any criteria to discard useless critical pairs, like Algorithm 1 represents, is not efficient at all. In the signature-based world there exist two main criteria to detect in advance a part of computations that are not necessary to be made:

1. The **non-minimal signature criterion** which is mainly based on the knowledge of syzygies respectively their leading monomials and comparing them with the signatures of critical pairs in question.

Algorithm 1 Generic signature-based Gröbner basis algorithm w.r.t. $<$ (SBA)

Input: $F = (f_1, \ldots, f_k)$ a finite sequence of elements in $\mathcal{R}$
Ensure: poly(G) a Gröbner basis for $\langle F \rangle$ w.r.t. $<$

```
1: G ← ∅, P ← ∅
2: for (i = 1, ..., k) do
3:    g_i ← (e_i, f_i)
4:    G ← G ∪ {g_i}
5:    P ← P ∪ {(g_i, g_j) | g_i, g_j ∈ G, j < i}
6: while (P ≠ ∅) do
7:    Let (f, g) ∈ P such that S(f, g) has minimal signature w.r.t. ≺.
8:    P ← P \ {(f, g)}
9:    Reduce S(f, g) sig-safe to r.
10:   if (poly(r) ≠ 0 and r is not sig-redundant to G) then
11:      P ← P ∪ {(r, h) | h ∈ G, S(r, h) not non-minimal}
12:      G ← G ∪ {r}
13: return poly(G)
```

2. The **rewritable signature criterion** which is based on the fact that one can detect relations between to be computed polynomials in advance by looking at the corresponding signatures.

For the focus of this paper we are neither interested in the specific variants these criteria can be implemented nor in a comparison of those in terms of efficiency and timings. It is enough to keep in mind that both criteria are based on a comparison between the signature of labeled polynomials considered during the algorithm's workings. Our point of interest is the connection between purely polynomial data to the signatures, a situation taking place whenever a labeled polynomial is multiplied by a term from $\mathcal{R}$.

The best-known and most efficient implementations of signature-based Gröbner basis algorithms today are

1. Faugère's F5 Algorithm ([11]) and optimizations ([9, 8, 10]),
2. Gao, Guan and Volny's G2V Algorithm ([13]),
3. Arri and Perry's algorithm ([1]), and
4. Gao, Volny and Wang's GVW Algorithm ([14]).

Whereas the first three mainly differ in their usage and implementation, i.e. aggressiveness, of the above mentioned signature-based criteria to detect useless critical pairs, they share $\prec_{\text{pot}}$ as the extended ordering used on the signatures. This leads to an incremental behaviour of all those algorithms, i.e. given $F = (f_1, \ldots, f_m)$ as input they all compute a Gröbner basis poly(G_2) of $\langle f_1, f_2 \rangle$, then a Gröbner basis poly(G_3) of $\langle f_1, f_2, f_3 \rangle$, and so on until the last iteration step computes poly$(G_m) =$ poly(G), a Gröbner basis of $\langle F \rangle$.

However GVW is capable of using different orderings on the signatures. In [14] the authors show that the Schreyer ordering $\prec_{\text{s}}$ turns out to be the most efficient one for a wide range of example classes. Due to the fact that $\prec_{\text{s}}$ does not favour the position of the module element over the corresponding term a

non-incremental computation is achieved. In examples with lots of generating elements it is quite an advantage to use all generators of the input during the computations at a time instead of successively adding new information in each iteration step only.

Remark 2.7. *Let us have a closer look at some, at a first glance, strange points in Algorithm 1 for using it as a generic setting for* **most of** *the above mentioned implementations.*

1. *If we assume $\prec = \prec_{\text{pot}}$, then Algorithm 1 clearly computes a Gröbner basis of $\langle F \rangle$ in an incremental fashion, storing the critical pairs of higher index in P, too, but prolonging their reduction until all elements of lower index have been processed.*

2. *If there exist several critical pairs in P of the same signature in Line 7, choose the one that entered P first.*

For a shorter notation in the following we often speak about s-polynomials in P. It is clear that we always mean the s-polynomial of the corresponding critical pair in P.

As already mentioned above, the crucial points for further investigating the algorithms' behaviour for inhomogeneous input data are all happening during the handling of a single s-polynomial:

1. Generating the s-polynomial, and then

2. multiplying a corresponding reducer for this s-polynomial to execute a sig-safe reduction step.

It follows that thinking of incremental signature-based algorithms our discussion always takes place inside one iteration step. In other words, we no longer need to distinguish between incremental and non-incremental signature based algorithms for our further discussion.

Still, one problem is missing we have not taken into account until now, but which can be understood as the starting point for our research: The initial description of F5 in [11] does not completely fit into the generic frame of Algorithm 1.

3 F5, or the evil of presorting labeled elements based on purely polynomial data

In [11] the first presentation of Faugère's F5 Algorithm is given. There Faugère restricts the input of the algorithm to homogeneous data. None of the successors of F5, like G2V or GVW have this restriction. So what is the decisive factor here? The main point is that the correctness of signature-based Gröbner basis computations is based on the fact that all s-polynomials are handled by increasing signatures. Without this requirement incorrect results can follow, especially when using the signature-based criteria the efficiency of F5 and its relatives relies on.

So let us have a look why F5's behaviour cannot be covered completely by Algorithm 1: Instead of choosing the next to be considered s-polynomial from

P by minimal signature (Line 7), F5 uses a presorting of P by the degree of the corresponding s-polynomials. So for F5 one needs to change Algorithm 1 beginning in Line 7:

Algorithm 2 Presorting changes for F5

```
1: ...
2: d ← min {deg(f, g) | (f, g) ∈ P}
3: Q ← {(f, g) | deg(f, g) = d}
4: P ← P\Q
5: while (Q ≠ ∅) do
6:    Let (f, g) ∈ Q such that S(f, g) has a minimal signature w.r.t. ≺.
7:    Q ← Q\ {(f, g)}
8:    ...
```

Of course, changing Algorithm 1 as explained above is not enough to ensure the correctness of the resulting algorithm. Replacing the corresponding parts of Algorithm 1 with the pseudo code of Algorithm 2 we need to distinguish where to store newly computed s-polynomials once an element r with $\mathrm{poly}(r) \neq 0$ is achieved. This postsorting is explained in Algorithm 3:

Algorithm 3 Postsorting changes for F5

```
1: ...
2: if (poly(r) ≠ 0 and r is not sig-redundant to G) then
3:    P ← P ∪ {(r, h) | h ∈ G, S(r, h) not non-minimal, deg(r, h) > d}
4:    Q ← Q ∪ {(r, h) | h ∈ G, S(r, h) not non-minimal, deg(r, h) = d}
5:    G ← G ∪ {r}
6: ...
```

Due to the fact that only sig-safe reductions are executed it is quite possible that $\deg(\mathcal{S}(r,h)) = d$, even in the homogeneous case. In this situation $\mathcal{S}(r,h)$ corresponds to a not sig-safe reduction step of r.

Remark 3.1. *Note that in [11] the problem of a reduction with an element of higher signature is solved in a slightly different way: Once such a reduction is noticed, the corresponding s-polynomial of higher signature is generated, and due to the fact of working with homogeneous data only, it is clear that this s-polynomial has the very same degree as the other elements already in Q. Thus it is directly added to Q, sorted in by increasing signature. The way we describe the algorithm here is equivalent, but easier to understand. See [10] for more information on this.*

Whenever one wants to compute a Gröbner basis for an inhomogeneous ideal with F5 the idea of homogenization can be used:

1. Homogenize the elements of F w.r.t. some new variable, call this sequence F^{h}.
2. Compute the Gröbner basis G^{h} for this homogeneous ideal $\langle F^{\mathrm{h}} \rangle$.
3. Cut down G^{h} to a Gröbner basis G for $\langle F \rangle$.

On the one hand, this attempt has the advantage to compute step-by-step intermediate Gröbner bases of s-polynomials up to a given degree d. Thus during the computations the degree of the elements investigated never drops, so a nearly optimal choice is made from the point of view of having all possible reducers available when they are needed. So in terms of our ongoing discussion of F5 it is impossible that an element in P will later on transform to a new labeled polynomial in G that could be useful for a reduction of an element currently in Q.

On the other hand, the problem of this approach is that computing a Gröbner basis for $\langle F^{\mathrm{h}} \rangle$ can be much harder than the computations in the inhomogeneous case. Thus it is desired to compute Gröbner bases for inhomogeneous input directly in signature–based algorithms, too.

What is really needed for the correctness of any signature–based algorithm? The question is pretty easy, one must process all critical pairs by increasing signatures, otherwise criteria checks may corrupt data, and wrong pairs are marked to be useless. Two questions arise from this investigation:

1. Does F5 compute new elements by increasing signatures throughout the algorithm's working assuming homogeneous input?

2. If the answer to the first question is positive, does this also hold using inhomogeneous input?

Let us try to answer these questions:

1. To answer the first question we need to find a connection between the degree of a labeled polynomial (i.e. the degree of the polynomial part of it) and its signature.

   Again, from a purely polynomial point of view, constructing s-polynomials of homogeneous polynomials has some nice properties: Let f and g be two homogeneous polynomials in $\mathcal{R}$. Computing corresponding multipliers u and v such that $u\,\mathrm{lt}(f) = v\,\mathrm{lt}(g)$ we can construct their s-polynomial $\mathcal{S}(f,g)$. It clearly holds that $\deg(u\,\mathrm{lt}(f)) = \deg(v\,\mathrm{lt}(g))$. As both $\mathrm{poly}(f)$ and $\mathrm{poly}(g)$ are homogeneous, $u\,\mathrm{poly}(f)$ and $v\,\mathrm{poly}(g)$ are homogeneous, too. Thus for all monomials t in the support of f it holds that

   $$\deg(ut) = \deg(u\,\mathrm{lt}(f)) = \deg(u) + \deg(f).$$

   A similar statement holds for the monomials of g. It follows that

   $$\deg(\mathcal{S}(f,g)) = \deg(u\,\mathrm{lt}(f)) = \deg(v\,\mathrm{lt}(g)) = \deg(f,g).$$

   With this in mind let us see how the signature and the degree of a labeled polynomial, with homogeneous polynomial part, computed in F5 are related to each other..

   (a) In the beginning the labeled polynomials $g_i = (e_i, f_i)$ are initialized for $i \in \{1, \ldots, m\}$. In this situation we know that

   $$\text{sig-deg}(g_i) = \deg(g_i).$$

   Note that this even holds if $\mathrm{poly}(g_i)$ is inhomogeneous.

(b) The first s-polynomials are generated by g_i and elements g_j, $j < i$. Again, let u and v be the corresponding multipliers such that $u\,\mathrm{lt}(g_i) = v\,\mathrm{lt}(g_j)$. W.l.o.g. we can assume that $\mathrm{sig}(ug_i) = ue_i \succ ve_j = \mathrm{sig}(vg_j)$. Since g_i and g_j are both homogeneous the degree of the critical pair (g_i, g_j) and $\mathcal{S}(g_i, g_j)$ is the same. Moreover, reductions by homogeneous elements do not change the degree of the s-polynomial. Since those reductions are sig-safe, the signature does not change, too. Assuming that the sig-safe reduction of $\mathcal{S}(g_i, g_j)$ results in a labeled polynomial r with $\mathrm{poly}(r) \neq 0$, it holds that $\deg(r) = \deg(\mathcal{S}(g_i, g_j)) = \deg(ug_i)$. Besides this we know that $\mathrm{sig}(r) = ue_i$; in other words

$$\text{sig-deg}(r) = \deg(u\pi(e_i)) = \deg(ug_i) = \deg(r).$$

Thus all labeled polynomials f constructed in this way fulfill

$$\text{sig-deg}(f) = \deg(f).$$

(c) As a last step we have to take a look at labeled polynomials derived from s-polynomials $\mathcal{S}(f, g)$ generated by arbitrary elements f and g from G. For this let us assume again multipliers $u, v \in \mathcal{R}$ such that $u\,\mathrm{lt}(f) = v\,\mathrm{lt}(g)$. By the above discussion we can assume that

$$\text{sig-deg}(f) = \deg(f) \quad \text{and} \quad \text{sig-deg}(g) = \deg(g).$$

W.l.o.g. let $u\mathrm{sig}(f) \succ v\mathrm{sig}(g)$. As $\deg(r) = \deg(uf - vg) = \deg(uf)$, where r is the corresponding sig-safe reduced labeled polynomial,[4] we see that also in this situation equality of the polynomial degree and the signature degree holds:

$$\text{sig-deg}(r) = \deg(u) + \text{sig-deg}(f) = \deg(u) + \deg(f) = \deg(r).$$

We see that, assuming homogeneous input for a signature-based Gröbner basis algorithm, it is useless to presort the pair set P by increasing degree of the s-polynomial and later on sort Q by increasing signatures: The signature of an s-polynomial in Q is always smaller than the signature of any element left in P.

So the answer to the first question is *yes*, F5 computes by increasing signature if the input data is homogeneous.

2. Next, let us try the very same discussion, this time under the assumption that the underlying polynomial data is inhomogeneous. In this situation the connection between $\text{sig-deg}(f)$ and $\deg(f)$ of a labeled polynomial f becomes more complicated:

   (a) Clearly, for the initial labeled polynomials nothing changes: $g_i = (e_i, f_i)$ with

   $$\text{sig-deg}(g_i) = \deg(f_i) = \deg(g_i).$$

[4] Again we assume that $\mathrm{poly}(r) \neq 0$. Otherwise the statement is trivial.

(b) For (g_i, g_j) generated by initial elements $g_i, g_j \in G$ it still holds that the degree of the critical pair, $\deg(g_i, g_j)$, is equal to sig-deg(ug_i) where $u, v \in \mathcal{R}$ are the respective multipliers of g_i and g_j in the corresponding s-polynomial $\mathcal{S}(g_i, g_j)$. W.l.o.g. let us assume in the following that $\text{sig}(\mathcal{S}(g_i, g_j)) = \text{sig}(ug_i)$. Working with inhomogeneous data, computing the s-polynomial and sig-safe reducing it even further the polynomial degree can drop. So the reduced labeled polynomial r ($\text{poly}(r) \neq 0$) in the end only fulfills the much weaker inequality

$$\deg(r) \leq \text{sig-deg}(r) = \text{sig-deg}(ug_i) = \deg(ug_i) = \deg(g_i, g_j).$$

So from this point on for all newly computed labeled polynomials g the degree of the polynomial part of g can be smaller than the degree of its signature.

(c) This leads to the problem that at the moment we generate critical pairs (f, g) of labeled polynomials f and g, again assuming $u\text{sig}(f) \succ v\text{sig}(g)$ where $u, v \in \mathcal{R}$ are the respective multipliers of f and g for $\mathcal{S}(f, g)$,

$$\deg(f, g) < \text{sig-deg}(uf)$$

is possible. It is even way more likely than having an equality of those degrees. From this point onwards it is clear that there is for any labeled polynomial h no stronger connection between $\deg(h)$ and sig-deg(h) besides $\deg(h) \leq \text{sig-deg}(h)$.

From this we see that whereas it is still safe to choose critical pairs by increasing signature, F5's attempt to presort by the degree of the critical pairs can cause problems. Think about the following quite likely situation:[5]

Let (f, g) and (f', g') be two critical pairs in P, and let u, v respectively $u', v' \in \mathcal{R}$ be the respective multipliers for $\mathcal{S}(f, g)$ respectively $\mathcal{S}(f', g')$. Assume that $u\text{sig}(f) \succ v\text{sig}(g)$ and $u'\text{sig}(f') \succ v'\text{sig}(g')$. Moreover, assume that $\deg(uf) < \deg(u'f')$. In F5 this means that once all critical pairs of degree $< \deg(uf)$ have been processed, (f, g) is added to Q, whereas (f', g') stays in P and its further computation is postponed to a later point. In the situation of underlying, inhomogeneous polynomial data it is still possible that $u\text{sig}(f) \succ u'\text{sig}(f')$ as we have just seen. This would mean that an element of higher signature is computed *before* an element of lower signature.

The main problem is that computing by increasing signatures is of supreme importance in the signature-based world, correctness and termination of the different algorithms (using variants of the non-minimal signature criterion and the rewritable signature criterion) is based on this fact.

Thus the answer to the second question is *no*, using F5's presorting of critical pairs by their polynomial degree can lead to false computations if input data is inhomogeneous.

[5] For example, in `Eco-11` such a situation happens hundreds of times

So at the end of this discussion we arrive at some quite astonishing facts:

We have seen that in the case of homogeneous input F5's presorting of critical pairs is useless and does not change anything w.r.t. to the order in which the algorithm handles its critical pairs: Those are still processed by increasing signatures. On the other hand, exactly this presorting harms the correctness of F5 (and of course any other signature-based algorithm with such a presorting implemented) once it comes to inhomogeneous input data. Thus we can state the following rule: One should always implement signature-based Gröbner basis algorithms without a purely polynomial degree preselection due to the fact that this method

1. does not change any computational aspect in the homogeneous case, and
2. enables to compute Gröbner bases for inhomogeneous input.

It follows that we can remove this pre- and postsorting mechanisms (Algporithm 2 resp. Algorithm 3) in F5 without changing any computational step at all. This means that from this point on we can also assume F5 as a variant of Algorithm 1.

Remark 3.2. *Due to the equality between the polynomial degree and the degree of a signature for homogeneous elements signature-based algorithms are designed to handle Gröbner basis computations in this setting very well, discarding lots of useless critical pairs and sorting them by (polynomial) degree which is, in general, the best possible selection strategy in this case.*

4 A second cube of sugar, please

In the last section we have seen that in the homogeneous case signature-based algorithms choose the order in which critical pairs are handled rather optimal. Moreover, we modified F5 to ensure correct computations of all signature-based algorithms for inhomogeneous input, too. Thus the question about the algorithms' usefulness in the inhomogeneous comes to one's mind.

In [16] Giovini, Mora, Niesi, Robbiano, and Traverso introduce the notion of the so-called *sugar degree* of a polynomial. Later on, Bigatti, Caboara, and Robbiano describe the idea of a self-saturating variant of Buchberger's algorithm in [3]. There an in-depth discussion on the theoretical background of the sugar degree is given. The main idea behind this kind of degree is to improve Gröbner basis computations for inhomogeneous input by giving it the flavour of a homogeneous computation. Also there exist some other concepts for optimizations in the inhomogeneous setting, see for example [22] or the idea of self-saturation given in [3][6], the approach using the sugar degree is so far the most popular and most efficient one.

Definition 4.1. *Let $F = (f_1, \ldots, f_m)$ be a sequence of polynomials in $\mathcal{R}$. Assume F to be the input of a purely polynomial Gröbner basis algorithm computing G. Let f and g be two elements in G, and let $t \in \mathcal{M}$.*

1. *Then we define the* **sugar degree** *in the following way:*

[6] see also Section 6

(a) $\text{s-deg}(f_i) := \deg(f_i)$ *for all* $i \in \{1, \ldots, m\}$,

(b) $\text{s-deg}(tf) := \deg(t) + \text{s-deg}(f)$, *and*

(c) $\text{s-deg}(f + g) := \max\{\text{s-deg}(f), \text{s-deg}(g)\}$.

2. *For a critical pair* (f, g) *we define the sugar degree by the sugar degree of the corresponding s-polynomial:*

$$\text{s-deg}(f, g) := \text{s-deg}\left(\mathcal{S}(f, g)\right).$$

Clearly, if F consists of homogeneous polynomials the sugar degree coincides with degree throughout the Gröbner basis computation. When computing with inhomogeneous data, the sugar degree becomes a useful tool: The sugar degree mimics the degree the elements, considered in a Gröbner basis computation, would have, if the input sequence would have been homogenized in the very beginning. Thus using the sugar degree the following threepartite sorting of critical pairs emerged to be very efficient in a wide class of example sets tested (see [16] for more information on this):

1. increasing sugar degree,
2. increasing degree,
3. increasing w.r.t. $<$.

Doing this, critical pairs are sorted as in the homogeneous situation without the overhead of homogenizing at all. Clearly, this sorting still needs not be optimal, but it has a rather positive influence on the efficiency of Gröbner basis algorithms in general.

Lying a focus on signature-based Gröbner basis algorithms the question of how their sorting of critical pairs by increasing signatures is related to the above presented "sugared" ordering needs to be clarified. The nice fact is that both coincide quite a lot.

Theorem 4.2. *The degree of the signature of a labeled polynomial* f*, computed in a signature-based Gröbner basis algorithm, coincides to the sugar degree of the polynomial part of* f*, that is*

$$\text{sig-deg}(f) = \text{s-deg}\left(\text{poly}(f)\right).$$

Proof. Let f, g be two labeled polynomials computed in a signature-based Gröbner basis algorithm. Moreover, let $F = (f_1, \ldots, f_m)$ be the input sequence of polynomials in $\mathcal{R}$.

1. For each such f_i it holds that the initial labeled polynomial $g_i = (e_i, f_i)$ fulfills that

$$\text{sig-deg}(g_i) = \deg(g_i) = \deg(f_i) = \text{s-deg}(f_i).$$

2. For any term $t \in \mathcal{R}$ it holds that $\text{sig-deg}(tf) = \deg(t) + \text{sig-deg}(f)$.

3. Let u and v be multipliers in $\mathcal{R}$ such that $u\,\text{lt}(f) = v\,\text{lt}(g)$. Note that by our previous discussion in Section 2 we can assume the s-polynomial to be not non-minimal. W.l.o.g. let $\text{sig}(uf) \succ \text{sig}(vg)$. Then it holds that

$$\text{sig-deg}\left(f, g\right) = \text{sig-deg}\left(\mathcal{S}(f, g)\right) = \text{sig-deg}(uf).$$

Thus the signature degree of a labeled polynomial f in a signature-based Gröbner basis algorithm coincides with the sugar degree of the corresponding polynomial part $\mathrm{poly}(f)$. □

This means that any signature-based algorithm with underlying monomial order $<$ prefering the polynomial degree computes new elements for the Gröbner basis w.r.t. the sugar degree. Let us have a closer look at different possible situations depending on the chosen monomial ordering:

1. Assume a degree-prefering ordering $<$ on $\mathcal{M}$.
   (a) Using $\prec_{\mathrm{s}}$ on the signatures, this leads to a non-incremental signature-based algorithm choosing critical pairs by increasing sugar degree. This is based on the fact that
   $$\mathrm{sig}(f) \prec_{\mathrm{s}} \mathrm{sig}(g) \Longleftrightarrow \text{sig-deg}(f) < \text{sig-deg}(g).$$
   (b) On the other hand, if we choose $\prec_{\mathrm{pot}}$ as ordering on the signatures the situation gets a bit more complicated: The algorithm prefers the signatures of higher module position. Assuming a critical pair being generated by two elements with signatures of different module position it is not clear that those signature also has a higher degree than the one of lower module position. We have seen in various tests that ordering by increasing sugar degree in such a setting even breaks the incremental structure of the algorithm, for example in `Cyclic-5`.
2. If a monomial ordering $<$ is given that is possibly not degree-prefering then sorting critical pairs by increasing signature need not lead to a pair set sorted by increasing sugar degree. Even if we use $\prec_{\mathrm{s}}$ on the signatures we cannot guarantee such a behaviour.

So the question arises if we can relax the restriction of sorting critical pairs by increasing signatures to a sorting by increasing sugar degree and hopefully get a more efficient algorithm? Sadly, until now, we are not able to plainly sort critical pairs by increasing sugar degree. Implementing such an ordering on the critical pairs one gets wrong Gröbner basis computations in many cases. Moreover, even if the computations lead to correct results the algorithm lacks quite a lot of its efficiency. Clearly, the strength of signature-based criteria relies on the fact that critical pairs are computed by increasing signatures in order to take advantage of a maximal possible set of smaller signatures to detect useless elements.

Furthermore, as we have seen in Section 3, even if the signature-based algorithm presorts the critical pairs by increasing sugar degree, a refined ordering w.r.t. the polynomial degree (as in the purely polynomial setting) cannot be done in the signature-based world, but the signature ordering $\prec$ has to be prefered towards the polynomial ordering $<$. Nevertheless we can conclude that signature-based Gröbner basis algorithms choose by default a rather good selection strategy, both in the inhomogeneous and in the homogeneous case. This choice is kind of optimal if we assume a degree-prefering monomial ordering $<$ on $\mathcal{M}$ and $\prec_{\mathrm{s}}$ on the signatures.

5 Still, there is a sour taste left

The discovery of the last section seems to be compatible with published experimental results. There we have seen that sometimes signature-based algorithms have problems computing Gröbner bases of inhomogeneous ideals, for example computing `Eco-11` is way harder for F5 than computing `Eco-11-h` w.r.t. the graded reverse-lexicographical ordering. For `Eco-11` we have seen that switching from incremental to non-incremental computations this can improve the timings quite a lot.[7] Still it is not clear that such a solution always works.

In Buchberger-like algorithms the problem is often just the other way around. Homogenizing ideals and trying to compute a corresponding Gröbner basis can be a much harder problem than the computations in the inhomogeneous setting. So the question arises where exactly the problems of signature-based Gröbner basis algorithms lie w.r.t. inhomogeneous input?

- The selection strategy is efficient as we have seen in Theorem 4.2.
- Also the criteria detecting useless critical pairs work quite great in the inhomogeneous setting, discarding a lot more elements than an Gebauer–Möller implementation in most of the examples as shown in several publications on signature-based algorithms, see for example [10, 13, 14].
- So the only situation where problems can occur is the reduction process. In there we can ignore the reducers of lower index, since they are handled without any difference as in a Buchberger–like algorithm. So the reductions with current index labeled polynomials seem to be left as a potential source of trouble.

Let us investigate this case a bit more carefully: Due to the fact that we lose the connection

$$\deg(f) = \text{sig-deg}(f)$$

for all labeled polynomials f computed in signature-based algorithms when switching from homogeneous to inhomogeneous input, forcing the reduction to be sig-safe can have really bad impact on the algorithms behaviour. Whereas in the homogeneous case the signature of the multiplied reducer can only be greater due to either its signature index (considering $\prec_{\text{pot}}$) or lexicographic considerations (depending on the underlying monomial ordering $<$), it always holds that

$$\text{sig-deg}(f) = \text{sig-deg}(tg)$$

where f is the element to be reduced, and $t \in \mathcal{R}$ the corresponding multiplier for the reducer $g \in G$. Assuming the input to be inhomogeneous it is even possible that a multiplied reducer has a signature of higher degree than the element to be reduced. Thus a lot more reductions could be discarded (not sig-safe) due to a higher signature of the multiplied signature than this happens in the homogeneous setting. This again means that a bunch of new critical pairs are generated and tested and computed. But this time the signatures of these critical pairs need not have the same degree. Let us give an example: Assume a labeled polynomial f to be reduced by another labeled polynomial $g \in G$, i.e. there exists a term $t \in \mathcal{R}$ such that $\text{lt}(f) = t\,\text{lt}(g)$. The reduction itself is not

[7] See Table 1 and [10].

allowed as $t\text{sig}(g) \succ \text{sig}(f)$. So in the following a new critical pair (tg, f) with signature $t\text{sig}(g)$ is generated and later on computed. The problem is the "later on": Whereas (g, f) has the same signature degree as f in the homogeneous setting, assuming polynomials to be inhomogeneous it is possible that

$$\text{sig-deg}(g, f) = \text{sig-deg}(tg) > \text{sig-deg}(f).$$

This means that the corresponding data needed from the reduction step of f and tg cannot be used in the algorithm at the time it really is needed. This triggers other reductions that would be helpful to take place at an earlier point of the algorithm to be delayed. All in all, correctness is still ensured, but the overhead that is computed due to all these not allowed and postponed reduction steps can have a clear penalty on the performance of signature-based Gröbner basis algorithms.

Luckily it seems that such bad behaviour happens not too often. We have tested over a rather wide class of benchmarks from [5] and [17]. Those cover different admissible orderings, finite and infinite ground fields, including parameters, etc. We have implemented four different variants of signature-based Gröbner basis algorithms in the computer algebra system SINGULAR:

1. The incremental F5 Algorithm including the optimizations mentioned in [9, 10] using $\prec_{\text{pot}}$ as ordering on the signatures.
2. Moreover, we have implemented a non-incremental Gröbner basis algorithm using the signature-based criteria to discard useless critical pairs presented in [11] using $\prec_{\text{s}}$ as ordering on the signatures.
3. Furthermore, we have two variants of the algorithm presented in [1], which we denote, for a shorter notation, AP:[8]
   (a) The first version uses $\prec_{\text{pot}}$ as ordering on the signatures and is thus an incremental algorithm.
   (b) The second implementation uses $\prec_{\text{s}}$ on the signatures in order to achieve non-incremental computations.

Remark 5.1. *Note that* G2V *respectively* GVW *is not competitive to the above mentioned 4 implementations in terms of performance as shown in [10]. This is due to fact that* G2V *as well as* GVW *lack a real implementation of the rewritable signature criterion, whereas* F5 *(respectively its various optimizations) and* AP *include rather aggressive implementations of it. Due to this we have waived to compare these variants of signature-based Gröbner basis algorithms from our investigations in order to not distort the results.*

Since the algorithms are still experimental, undergoing further development, they are currently not part of the stable SINGULAR repository. Still, they are publicly available in the branch `sba` at

`https://github.com/ederc/Sources`.

This branch is based on the current SINGULAR 3-1-4 developer version.

In the following, when speaking of *variants of* SBA we always refer to the 4 above mentioned implementations, and specify notation whenever needed.

[8] Refering to the authors of [1], Alberto Arri and John Perry

In this paper we do not focus on timings in detail, but we want to get a better feeling for the behaviour of signature-based Gröbner basis algorithms for inhomogeneous input. For this it is important to compare with computations for the corresponding homogenized data. In more detail, we need to have a look at the ratio between executed reduction steps and detections of higher signatures during the reduction process.

Remark 5.2. *There are several important observations from our experiments:*

1. *In a wide range of examples the number of sig-safe reduction steps is factor of* 1000 *greater than the number of higher signature detections. Examples are the* `Noon-n` *or* `Katsura-n` *benchmarks. So not depending on whether the input is homogeneous or not the influence of not sig-safe data is not even measureable.*

2. *Clearly, the number of higher signature detections one gets also depends on the order in which the list of possible reducers is searched through. As a first experience we can state that in most benchmarks, again independent of the homogeneity of the input polynomials, using the settings and heuristics of* Singular*'s internal, Gebauer-Möller-like Gröbner basis algorithm* `groebner` *is quite the best choice. Of course there are examples like* `Fabrice-24` *where adjusting the search by hand leads to an improvement in timings of a factor of* 10*, but in other examples exactly this choice of searching for reducers slows down computations by a factor of* 100 *and even more. So there is a lot of work to be done considering good heuristics for searching in the set of reducers; doing this by increasing respectively descreasing signature is not a good choise in a wide range of example classes.*

3. *The 4 different implementations behave totally differently in lots of examples. Sometimes* AP*'s implementation of the rewritable signature criterion is way better than* F5*'s, in other situations it is just the other way around. Moreover, finding a heuristic when to prefer incremental computations over non-incremental ones (for example for* `Katsura-n`*) is an area of great importance.*

In Figure 1 we give an overview of the behaviour of the 4 variants of Sba in several different benchmark sets, both homogeneous and inhomogeneous. We lay our focus on the differences in the reduction process with a look at the ratio between the number of higher signature detections and the number of reduction steps in total. We give these ratios in percentage, represented by the height of the respective bars in the diagrams. With this we would like to get a better feeling for the influence of losing the connection between $\deg(f)$ and sig-$\deg(f)$ in the inhomogeneous setting.

We have tested the algorithms with a wide range of benchmarks. Due to the fact that we must restrict ourselves not overcharging the reader, we present in this paper results for a part of our test suite. Those benchmarks represent the algorithms' overall behaviour quite accurately. The reader interested in the complete data set can get it at

`https://github.com/ederc/benchmarks`.

We have 4 variants of SBA to be tested for homogeneous and for inhomogeneous input each, thus we must represent 8 different values per benchmark. We decided to visualize our results by colorized bars: We chose blue, green, red, and orange as colors for the 4 variants of SBA:

1. Blue stands for F5 with $\prec_{\text{pot}}$.
2. Green is used for AP with $\prec_{\text{pot}}$

So blue and green visualize incremental computations in the following.

3. Results for F5 using $\prec_{\text{s}}$ are given in orange.
4. Red represents data processed by AP equipped with $\prec_{\text{s}}$.

Thus the warm colors represent computations done in a non-incremental way. In this color scheme the darker variation illustrates the results for the respective inhomogeneous example, whereas the lighter variation stands for results achieved computing the corresponding homogenized example.

What we can see having a closer look at the diagrams in Figure 1 is that we cannot come to any accurate and definite decision: Sometimes the ratio is several times greater in the homogeneous setting than in the inhomogeneous one (see, for example, `Cyclic-8`[9] and `Ext-Cyclic-6` for F5 and AP using $\prec_{\text{pot}}$). On the other hand, in examples like `Ilias-12` it is just the other way around.

Talking about incremental versus non-incremental computations there is an inclination that the ratio of the number of higher signature detections and the number of reduction steps done is mostly smaller in the non-incremental setting. At least there seem to be no such amplitudes as we can see for the incremental computations, for example, in `Cyclic-8` or `Red-Eco-12`. Still one needs to keep in mind that we only present the ratios: For example, in `Katsura-12` the non-incremental variants of SBA are multiple times slower than the incremental ones, they do approximately 50 times more reduction steps. Nevertheless due to this high amount of reductions the ratio gets lower. Keep in mind that we lay our focus on the behaviour of signature-based Gröbner basis algorithms for inhomogeneous input, not on differences in performance for the respective different implementations.

To get a better feeling for the complete picture combine the ratios of Figure 1 with the timings given in Table 1. All examples where computed on a machine with an INTEL® XEON® X5460 @ 3.16GHz processor, 64 GB of RAM, and 120 GB of swap space. We used the abvoe mentioned implementation from `sba`[10] on a 2.6.31–gentoo–r6 GNU/Linux 64–bit operating system.

Furthermore, note that all of the examples presented in Figure 1 are computed w.r.t. the graded reverse-lexicographical ordering. In the complete benchmark set you can get online there are, for example, also computations w.r.t. lexicographical orderings. Due to our discussion in Section 4 it is clear that computations w.r.t. monomial orderings not prefering the polynomial degree are rather inefficient in terms of sorting critical pairs. For example, this explains the bad performance of our implemented signature-based algorithms in `Cyclic-x`, `Eco-x`, and `Katsura-x` when it comes to inhomogeneous computations w.r.t. the

[9] In the following we always use the inhomogeneous notation of the benchmark in the text and omit the "`(-h)`" for a better readability.

[10] Based on commit `291021c19066befbcdd8a7d7626e5ddc2d421db4`.

Test case	F5,$\prec_{\text{pot}}$	AP,$\prec_{\text{pot}}$	F5,$\prec_{\text{s}}$	AP,$\prec_{\text{s}}$
Cyclic-7	1.330	1.260	1.840	2.660
Cyclic-7-h	1.180	1.140	1.820	2.630
Cyclic-8	468.260	540.860	314.970	184.900
Cyclic-8-h	387.890	508.190	307.000	186.780
Ext-Cyclic-6	157.590	129.340	13.420	16.770
Ext-Cyclic-6-h	662.380	4,006.960	10.260	14.090
Ilias-12	3,447.510	458.480	639.870	283.960
Ilias-12-h	4,381.080	2,240.890	553.180	239.490
Eco-10	45.610	2.780	7.990	7.190
Eco-10-h	14.990	13.280	3.660	4.290
Eco-11	2,398.970	29.810	163.830	125.340
Eco-11-h	372.090	319.710	48.710	56.680
Red-Eco-11	2.620	4.850	14.530	19.470
Red-Eco-11-h	2.600	4.870	19.290	20.510
Red-Eco-12	22.010	37.270	161.530	386.340
Red-Eco-12-h	21.390	38.660	246.470	392.460
F-744	1.550	0.740	0.430	0.450
F-744-h	2.090	1.470	0.360	0.380
F-855	50.670	27.200	122.670	96.080
F-855-h	133.470	65.980	48.600	48.930
Fabrice-24	101.900	72.250	113.710	108.300
Fabrice-24-h	121.900	101.190	361.570	326.040
Katsura-12	111.690	61.490	1,287.250	1,303.360
Katsura-12-h	109.970	54.510	1,260.380	1,223.710

Table 1: Timings in seconds for the computation of a Gröbner basis for the given test case.

lexicographical ordering. For not degree-prefering orderings it is more efficient to compute a Gröbner basis w.r.t. the graded reverse-lexicographical ordering, for example, and then to use Gröbner conversion via FGLM.

6 Conclusion and further research

In this paper we have given an in-depth discussion about the behaviour of signature-based Gröbner basis algorithms in the inhomogeneous case.

Firstly we solved the long open problem why F5, as initially presented by Faugère in [11], is restricted to homogeneous input data. We have not only seen how to change F5 to enable computations for inhomogeneous elements, but also shown that this modification even simplifies the whole algorithm. This makes it easier for a reader without prior knowledge of signature-based algorithms to get access to this branch of Gröbner basis theory.

Moreover, we have presented for the first time the strong connection between the signatures of labeled polynomials in signature-based Gröbner basis algorithms and the sugar degree of the corresponding polynomial parts. It is a delightful discovery that signature-based algorithms sort the corresponding pair set in a nearly optimal order from the polynomial point of view when assum-

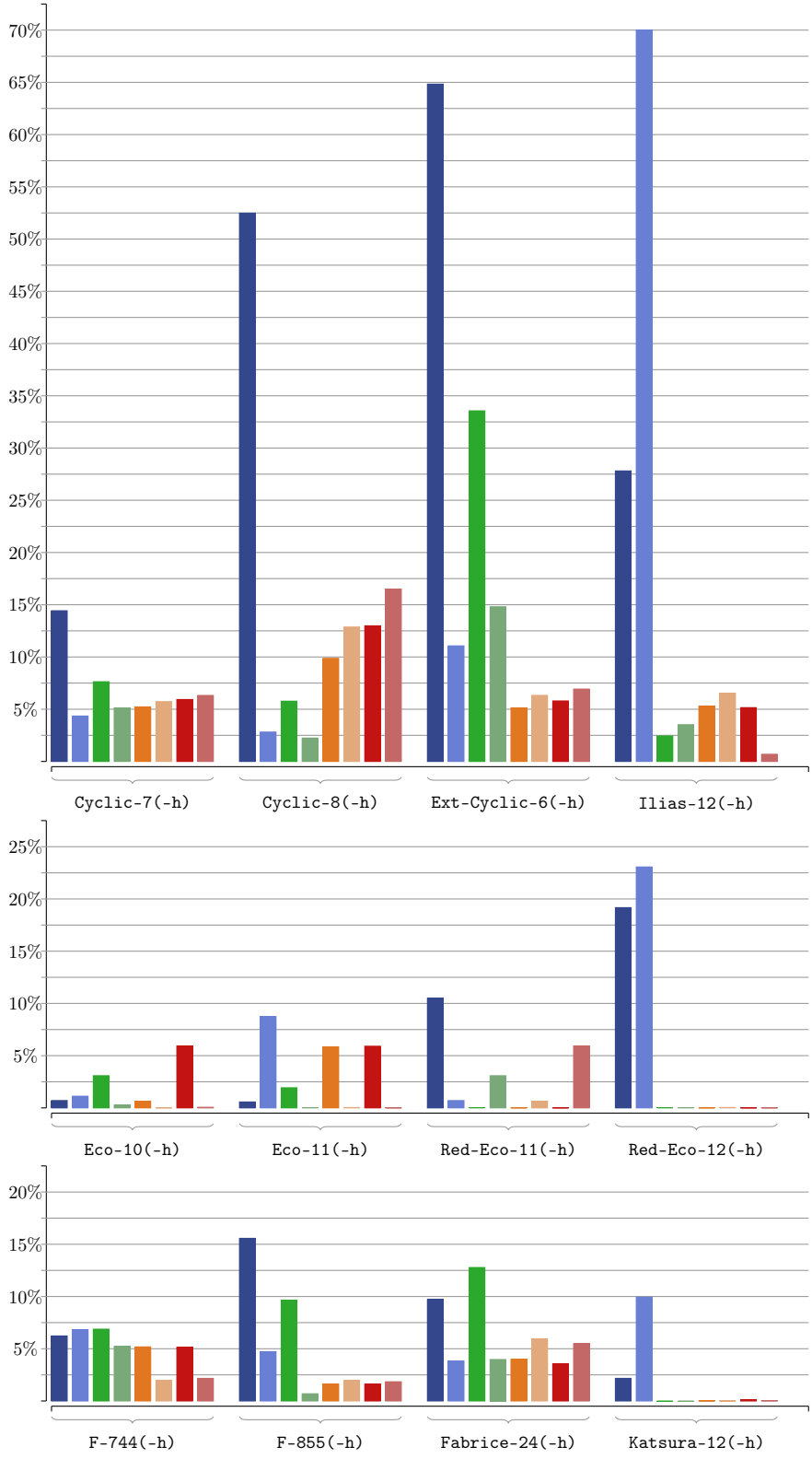

Figure 1: Ratio of higher-signature detections to reduction steps for various benchmarks

ing a degree-prefering monomial ordering. Still, trying to reorder critical pairs for example by increasing sugar degree in the situation a of using a monomial ordering not prefering the polynomial degree is bounded by the condition of computing by increasing signatures. The question if we can find better orderings on the signatures arises. Again, this is something which should be decided by a heuristic depending on various parameters of the given input.

As a last step we have tried to investigate how the lost connection between polynomial degree and signature degree in the inhomogeneous setting can have a bad influence on the sig-safe reduction process all signature-based algorithms are based on. The question of an increasing number of higher signature detections, and thus rejected possible reductions, come to one's mind in that case. Having computed a quite broad range of examples we have seen that there cannot be found an apparent disadvantage in using signature-based algorithms for the computation of inhomogeneous Gröbner bases. There are always some specific examples where the number of higher signature detections increase quite a lot comparing the homogeneous setting to the inhomogeneous one, but there are also examples where it is just the other way around.

A possible field of further investigations in the direction of inhomogeneous signature-based Gröbner basis computations could be to combine it with the idea of self-saturation given in [3]. The overall idea lies in the fact to use special kinds of reduction steps to achieve so-called *(weak) saturating remainders*. Thereby the Gröbner basis algorithm starts with the homogenized set of generators, but instead of plainly computing the homogeneous Gröbner basis of the homogenized input data, reducers respectively the remainder of a reduction are exchanged by saturated pendants. By this, a resulting set $G = \{g_1, \ldots, g_s\}$ is achieved such that $\{g_1^{\text{deh}}, \ldots, g_s^{\text{deh}}\}$[11] is a Gröbner basis of the inhomogeneous input data. Whereas the process of self-saturation has a positive effect on Buchberger-like Gröbner basis algorithms as shown in detail in [3], the restriction to sig-safe reductions in signature-based algorithms could put a too strong constraint on the freedom of choice for the saturated pendants. This needs a more in depth investigation which would need a specialized implementation of a self-saturating signature-based algorithm. This is out of the scope of this paper.

Having various, quite efficient variants of signature-based Gröbner basis algorithms implemented it is time to not only investigate for better criteria discarding useless critical pairs, but also to start research in the area of heuristics for classes of example sets: When is an incremental computation better than a non-incremental one? Which combination of already known variants of the criteria is most efficient in which setting? So the area of signature-based algorithms matures: Having proved correctness and termination[12] for those kind of Gröbner basis algorithms over the last years, making new open source implementations available, we are now able to dive deeper in the theory and get a better feeling for how these algorithms behave in different situations. Clearly, entering inhomogeneous computations the question of how signature-based algorithms can be used, and if so, how they perform, for local monomial orderings emerges, too.

[11] "deh" denotes the dehomogenization of the homogeneous elements g_i.

[12] Or at least found ideas how to ensure termination for F5, see [8, 2, 15].

References

[1] Arri, A. and Perry, J. The F5 Criterion revised. *Journal of Symbolic Computation*, 46(2):1017–1029, June 2011. Preprint online at `arxiv.org/abs/1012.3664`.

[2] Ars, G. *Applications des bases de Gröbner à la cryptographie*. PhD thesis, Université de Rennes I, 2005.

[3] Bigatti, A. M., Caboara, M., and Robbiano, L. Computing Inhomogeneous Gröbner Bases. *Journal of Symbolic Computation*, 46:498–510, 2011.

[4] Bigatti, A. M., La Scala, R., and Robbiano, L. Computing toric ideals. *Journal of Symbolic Computation*, 27:351–365, 1999.

[5] Bini, D. A. and Mourrain, B. Polynomial Test Suite. 2012. `http://www-sop.inria.fr/saga/POL/`.

[6] Buchberger, B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 1965.

[7] Decker, W., Greuel, G.-M., Pfister, G., and Schönemann, H. SINGULAR *3-1-4 — A computer algebra system for polynomial computations*, 2012. `http://www.singular.uni-kl.de`.

[8] Eder, C., Gash, J., and Perry, J. Modifying Faugère's F5 Algorithm to ensure termination. *ACM SIGSAM Communications in Computer Algebra*, 45(2):70–89, 2011. `http://arxiv.org/abs/1006.0318`.

[9] Eder, C. and Perry, J. F5C: A Variant of Faugère's F5 Algorithm with reduced Gröbner bases. *Journal of Symbolic Computation, MEGA 2009 special issue*, 45(12):1442–1458, 2010. `dx.doi.org/10.1016/j.jsc.2010.06.019`.

[10] Eder, C. and Perry, J. Signature-based Algorithms to Compute Gröbner Bases. In *ISSAC 2011: Proceedings of the 2011 international symposium on Symbolic and algebraic computation*, pages 99–106, 2011.

[11] Faugère, J.-C. A new efficient algorithm for computing Gröbner bases without reduction to zero F5. In *ISSAC'02, Villeneuve d'Ascq, France*, pages 75–82, July 2002. Revised version from `http://fgbrs.lip6.fr/jcf/Publications/index.html`.

[12] Faugère, J.-C., Gianni, P. M., Lazard, D., and Mora, T. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[13] Gao, S., Guan, Y., and Volny IV, F. A New Incremental Algorithm for Computing Groebner Bases. *Journal of Symbolic Computation – ISSAC 2010 Special Issue*, 1:13–19, 2010.

[14] Gao, S., Volny IV, F., and Wang, D. A new algorithm for computing Groebner bases. 2010.

[15] Gash, J. M. *On efficient computation of Gröbner bases*. PhD thesis, University of Indiana, Bloomington, IN, 2008.

[16] Giovini, A., Mora, T., Niesi, G., Robbiano, L., and Traverso, C. "One sugar cube, please" or selection strategies in the Buchberger algorithm. In *ISSAC'91*, pages 49–54, 1991.

[17] Gräbe, H.-G. *The SymbolicData Project – Tools and Data for Testing Computer Algebra Software*, 2011. `http://www.symbolicdata.org` .

[18] Greuel, G.-M. and Pfister, G. *A* SINGULAR *Introduction to Commutative Algebra*. Springer Verlag, 2nd edition, 2007.

[19] Huang, L. A new conception for computing Gröbner basis and its applications. , abs/1012.5425, 2010.

[20] Sun, Y. and Wang, D. A New Proof of the F5 Algorithm. *CoRR*, abs/1004.0084, 2010.

[21] Sun, Y. and Wang, D. A generalized criterion for signature related Gröbner basis algorithms. In *ISSAC 2011: Proceedings of the 2011 international symposium on Symbolic and algebraic computation*, pages 337–344, 2011.

[22] Ufnarovski, V. On the Cancellation Rule in the Homogenization. *Computer Science Journal of Moldova*, 16(1):133–145, 2008.

[23] Wichmann, T. Der FGLM-Algorithmus: verallgemeinert und implementiert in SINGULAR. *Diploma thesis at the university of Kaiserslautern*, 1997.